Dominik Wawszczak
Student ID Number: 440014
Group Number: 1

Parameterized Algorithms
Assignment 2

Problem 1

If G is not connected, we can analyze each connected component independently, subtracting the combined sizes of all other components from k. Therefore, for the remainder of the solution, we assume that G is connected.

If there exists a subset of vertices X such that $|X| \leqslant k$ and $G \setminus X$ is a tree, then $\mathrm{tw}(G) \leqslant k+1$, as $\mathrm{tw}(G \setminus X) = 1$, where tw denotes treewidth. We will use the algorithm presented in the lecture, which runs in time $27^l \cdot l^{\mathcal{O}(1)} \cdot n^2$, where l is the target treewidth and n is the number of vertices. We apply it to G with $l = k+1$. The algorithm will yield one of two possible outcomes:

1. Confirmation that $\mathrm{tw}(G) > k+1$.

   In this case, we conclude that no such subset X exists.

2. A tree decomposition of width at most $4k+8$.

   The remainder of the solution focuses on this scenario.

We now proceed with dynamic programming, assuming the decomposition is nice. For any subtree H of the decomposition and its boundary ∂H, let $f : V(\partial H) \to \{0, \ldots, 4k+9\}$. Define $\mathrm{dp}_H(f)$ as the size of the maximum set $Y \subseteq V(H \setminus \partial H)$ satisfying the following conditions:

1. $H \setminus \big(Y \cup f^{-1}(0)\big)$ is a forest.
2. Each tree in $H \setminus \big(Y \cup f^{-1}(0)\big)$ contains at least one vertex from ∂H, unless ∂H is empty (a special case when H is the entire decomposition).

   Note that this implies there are at most $|V(\partial H)|$ trees in $H \setminus \big(Y \cup f^{-1}(0)\big)$.
3. For all $v, u \in V(\partial H) \setminus f^{-1}(0)$, v and u belong to the same connected component (tree) of $H \setminus \big(Y \cup f^{-1}(0)\big)$ if and only if $f(v) = f(u)$.

To compute dp_H, we consider the following cases:

1. $H = \emptyset$

   In this case, we simply set $\mathrm{dp}_H(\text{empty function}) = 0$.

2. $H = \mathrm{introduceVertex}(H', v)$

   Here, $\mathrm{dp}_H(f) = \max(\{\mathrm{dp}_{H'}(f') \ : \ f = f'[v \mapsto a] \ \wedge \ (f^{-1}(a) = \{v\} \ \vee \ a = 0)\})$. We use the notation $f[p \mapsto q]$ to denote the function g defined by:

$$g(x) \ = \ \begin{cases} q, & \text{if } x = p, \\ f(x), & \text{otherwise.} \end{cases}$$

3. $H = \mathrm{introduceEdge}(H', v, u)$

   In this case,

$$\mathrm{dp}_H(f) \ = \ \begin{cases} \mathrm{dp}_{H'}(f), & \text{if } f(u) = 0 \text{ or } f(v) = 0, \\ \max(\{\mathrm{dp}_{H'}(f') \ : \ f' \text{ is good}\}), & \text{otherwise,} \end{cases}$$

   where f' is *good* if it satisfies:

3.1 $f'(v) \neq 0$ and $f'(u) \neq 0$ and $f'(v) \neq f'(u)$.

3.2 For all $w \in V(\partial H)$, if $f'(w) \in \{f'(v), f'(u)\}$, then $f(w) = f'(v)$. Otherwise, $f(w) = f'(w)$.

If no good function f' exists for a given f, set $\mathrm{dp}_H(f) = -\infty$.

4. $H = \mathrm{forgetVertex}(H', v)$

   Here, $\mathrm{dp}_H(f) = \max(\{\mathrm{dp}_{H'}(f') + [a \neq 0] \ : \ a \in \{0, \ldots, 4k+9\} \wedge f' = f[v \mapsto a]\})$, where $[P]$ denotes the Iverson bracket, i.e., $[P] = 1$ if P is true, and $[P] = 0$ otherwise.

   Additionally, we must check whether $f^{-1}(a) = \emptyset$. If this condition holds, we set $\mathrm{dp}_H(f) = -\infty$, except when H represents the entire decomposition, in which case this check is omitted.

5. $H = \mathrm{merge}(H', H'')$

   In this case $\partial H' = \partial H''$, and we calculate dp_H as $\mathrm{dp}_H(f) = \mathrm{dp}_{H'}(f) + \mathrm{dp}_{H''}(f)$.

The answer is $\mathrm{dp}_T(\text{empty function}) \geqslant V(G) - k$, where T is the full decomposition.

The total time complexity is bounded by

$$27^{k+1} \cdot (k+1)^{\mathcal{O}(1)} \cdot n^2 + n^{\mathcal{O}(1)} \cdot \left((4k+10)^{4k+10}\right)^2.$$

Thus, this algorithm is FPT when parameterized by k, which completes the proof.

Problem 2

We apply the color-coding technique. Let $c = k(l-1)$ denote the maximum total number of vertices along the paths, excluding the start and finish vertices. We color each vertex (excluding the start and finish vertices) with one of c colors. For each $i \in \{1, \ldots, k\}$, let $\mathrm{dp}_i(v, C)$ be true if and only if there exists a path from s_i to v such that the set of vertex colors on this path equals exactly C, with no two vertices sharing the same color. Here, we only consider subsets C of size at most $l-1$.

The values of dp_i are initially set to false and are then computed using the following rules:

1. $\mathrm{dp}_i(s_i, \emptyset) \ = \ \text{true}$.
2. $\mathrm{dp}_i(v, C) \ = \ \bigvee\limits_{(u,v) \in E(D)} (\mathrm{color}(v) \in C \wedge \mathrm{dp}_i(u, C \setminus \{\mathrm{color}(v)\}))$, for all $v \in V(D) \setminus \bigcup\limits_{j=1}^{k} \{s_j, t_j\}$.
3. $\mathrm{dp}_i(t_i, C) \ = \ \bigvee\limits_{(u,t_i) \in E(D)} \mathrm{dp}_i(u, C)$.

To compute these values correctly, ensuring that no dp entry is referenced before it has been calculated, subsets C are considered in increasing order of size. Subsequently, $\mathrm{dp}_i(v, C)$ is filled for all $v \in V(D)$.

The number of such subsets is bounded by:

$$\sum_{j=0}^{l-1} \binom{c}{j} \ \leqslant \ \sum_{j=0}^{l-1} c^j \ = \ \frac{c^l - 1}{c - 1} \ \leqslant \ c^l \ = \ (k(l-1))^l \ \leqslant \ (kl)^l.$$

Thus, this computation requires $\mathcal{O}\big((kl)^l k(n+m)\big)$ time, where $n = |V(D)|$, and $m = |E(D)|$.

The number of k-tuples of such subsets is bounded by $\left((kl)^l\right)^k = (kl)^{kl}$. We iterate over these tuples, performing the following for each $(C_1, \ldots, C_k)$:

1. Verify that the subsets $C_1, \ldots, C_k$ are pairwise disjoint.

   This step can be done in $\mathcal{O}(k^3 l)$ time.

2. Check whether $\mathrm{dp}_1(t_1, C_1) = \ldots = \mathrm{dp}_k(t_k, C_k) = \text{true}$.

   In this case, the answer to the problem is „Yes.".

The time complexity of one iteration is $\mathcal{O}\big((kl)^l k(n+m) + (kl)^{kl} k^3 l\big)$.

The probability of a single iteration failing to find a solution, assuming one exists, is $1 - \frac{c!}{c^c}$, as there are c^c ways to assign c colors to c vertices, and $c!$ of these avoid duplicate colors. Repeating the iteration e^c times reduces the failure probability to:

$$\left(1 - \frac{c!}{c^c}\right)^{e^c} < \left(1 - \frac{1}{e^c}\right)^{e^c} < 1 - \frac{1}{e},$$

a constant.

The total time complexity is:

$$\mathcal{O}\Big(e^{kl}\big((kl)^l k(n+m) + (kl)^{kl} k^3 l\big)\Big),$$

which is FPT when parameterized by $k + l$, as $kl \leqslant (k+l)^2$.

This algorithm can be determinized using an (n, c, c)-splitter, as discussed in the lecture. The splitter, with size $e^c c^{\mathcal{O}(\log c)}$, can be found in time $e^c c^{\mathcal{O}(\log c)} n \log n$. The resulting complexity is:

$$e^{kl}(kl)^{\mathcal{O}(\log kl)}\Big(n \log n + \mathcal{O}\big((kl)^l k(n+m) + (kl)^{kl} k^3 l\big)\Big),$$

which is also FPT when parameterized by $k + l$, completing the proof.